ENCODING METHOD OF DATA FIELD, EXTENDING METHOD OF INFORMATION FIELD AND COMPUTER SYSTEM

Patent Number: JP2001142694
Publication date: 2001-05-25
Inventor(s): DEBBAGE MARK
Applicant(s): HITACHI LTD
Requested Patent: ☐ JP2001142694
Application JP20000282611 20000918
Priority Number(s):
IPC Classification: G06F9/30; G06F9/32;
EC Classification:
Equivalents:

Abstract

PROBLEM TO BE SOLVED: To extend a range using a reservation bit under restriction that backward compatibility is maintained with the existing instruction set.
SOLUTION: This invention includes a method to extend an information field by a computer instruction with plural fields. One field of the computer instruction is provided with an operation code and other fields are respectively provided with the reservation bit 120. The reservation bit is combined with the information field in an extension field. By the combination, the information field with N bits is loaded on a first storage position of N+M bits and the reservation field with M bits is loaded on a second storage position of N+M bits. The information field in the first storage position is sign extended 116, the second storage position is shifted to left by N bits and shifted to zero at the right end. Exclusive OR operation 124 is implemented by the first storage position with the second storage position to direct to an field is guided. The result is the backward compatible extension field.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－142694
（P2001－142694A）
(43)公開日　平成13年５月25日(2001.5.25)

(51)Int.Cl.⁷		識別記号	FI			テーマコード(参考)
Ｇ０６Ｆ	9/30	３１０	Ｇ０６Ｆ	9/30	３１０Ｅ	５Ｂ０３３
	9/32	３５０		9/32	３５０Ｂ	
	9/34	３２０		9/34	３２０Ａ	

審査請求　未請求　請求項の数11　ＯＬ　（全 15 頁）

(21)出願番号　特願2000－282611(P2000－282611)

(22)出願日　平成12年９月18日(2000.9.18)

(31)優先権主張番号　０９／４１０６７５

(32)優先日　平成11年10月１日(1999.10.1)

(33)優先権主張国　米国（ＵＳ）

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　マーク・デバージ
イギリス国、ブリストル　ＢＳ41　９Ｄ
Ｕ、ロングアシュトン、チェリー・ロード
6

(74)代理人　100080001
弁理士　筒井　大和

Ｆターム(参考)　5B033 BA03 DC06 DC10

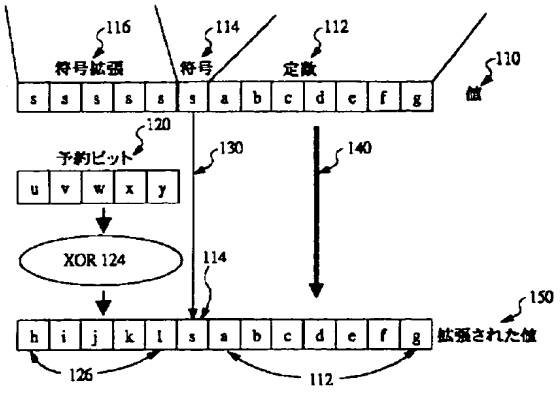
(54)【発明の名称】　データフィールドのエンコード方法、情報フィールドの拡張方法、及び、コンピュータシステム

(57)【要約】

【課題】　後方互換性が現存の命令セットと一緒に維持されるという制約で、予約ビットを使った範囲を拡張する要求がある。

【解決手段】　本発明は、複数のフィールドを有するコンピュータ命令で、情報フィールドを拡張する方法を含む。コンピュータ命令の１つのフィールドはオプコードを有し、別のフィールドは予約ビット１２０を有する。この予約ビットは拡張フィールド内で情報フィールドと組合される。その組合せは、Ｎビットの情報フィールドをＮ＋Ｍビットの第１記憶位置にロードし、Ｍビットの予約フィールドをＮ＋Ｍビットの第２記憶位置にロードする。第１記憶位置内の情報フィールドは符号拡張１１６され、第２記憶位置はＮビットだけ左にシフトされ、右端でゼロにシフトされる。第１記憶位置が第２記憶位置と一緒に排他的論理和１２４をとられて拡張フィールドを導く。結果は後方互換な拡張フィールドである。

図１



【特許請求の範囲】
【請求項１】　複数のフィールドを有する第１コンピュータ命令で、データフィールドをエンコードする方法であって、
前記命令内に任意の未使用データビットが存在するか否かを決定し、
前記未使用ビットのサブセットを使って前記データフィールドを拡張データフィールド内に拡張し、
複数のフィールドを有し、前記拡張データフィールドを使った第２コンピュータ命令が、前記データフィールドを用いた前記第１コンピュータ命令と後方互換であることを特徴とするデータフィールドのエンコード方法。
【請求項２】　請求項１記載の方法であって、
前記第２コンピュータ命令が、前記第１コンピュータ命令と同じ演算コードを含むことを特徴とするデータフィールドのエンコード方法。
【請求項３】　請求項１記載の方法であって、
前記未使用ビットが予約ビットを含むことを特徴とするデータフィールドのエンコード方法。
【請求項４】　請求項１記載の方法であって、
前記データフィールドが即値オペランドであることを特徴とするデータフィールドのエンコード方法。
【請求項５】　請求項１記載の方法であって、
前記データフィールドがディスプレースメントであることを特徴とするデータフィールドのエンコード方法。
【請求項６】　請求項１記載の方法であって、
前記サブセットが適当なサブセットであることを特徴とするデータフィールドのエンコード方法。
【請求項７】　複数のフィールドを有するコンピュータ命令で、情報フィールドを拡張する方法であって、
オプコードを含んだ前記複数のフィールドのうち第１フィールドを評価し、
もし前記複数のフィールド内に少なくとも１つの予約フィールドが存在するならば、拡張フィールド内に前記予約ビットと前記情報フィールドとを組合せ、
前記オプコードによって示された演算内で前記拡張フィールドを使用することを特徴とする情報フィールドの拡張方法。
【請求項８】　請求項７記載の方法であって、
前記拡張フィールドが前記情報フィールドと後方互換であることを特徴とする情報フィールドの拡張方法。
【請求項９】　請求項７記載の方法であって、
前記情報フィールドが、即値オペランドとディスプレースメントとを含んだグループから選択されることを特徴とする情報フィールドの拡張方法。
【請求項１０】　請求項７記載の方法であって、
前記組合せが、長さＮビットである前記情報フィールドを、第１記憶位置にロードし、
長さＭビットである前記予約フィールドを、第２記憶位置にロードし、
前記第１記憶位置内に前記情報フィールドを符号拡張し、
前記第２記憶位置をＮビットだけ左にシフトし、
前記第１記憶位置を前記第２記憶位置と一緒にビットワイズで排他的論理和をとって（Bitwise exclusive ORing）前記拡張フィールドを引き出し、
前記第１及び第２記憶位置が少なくともＭ＋Ｎビットを有することを特徴とする情報フィールドの拡張方法。
【請求項１１】　コンピュータ命令で情報フィールドを拡張するためのコンピュータシステムであって、
記憶装置と、プロセッサとを備え、
前記記憶装置が複数の記憶位置を含み、第１記憶位置が前記コンピュータ命令の前記情報フィールドを有し、第２記憶位置が前記コンピュータ命令の予約ビットを有し、
前記プロセッサが、前記第１記憶位置内に前記情報フィールドを符号拡張する符号拡張装置と、
前記第２記憶位置内で前記予約ビットを左にシフトするシフタと、
前記第１記憶位置と前記第２記憶位置とを組合せる論理ＸＯＲゲートとを有することを特徴とするコンピュータシステム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、一般にコンピュータ命令セットアーキテクチャに関し、特に、現存する情報フィールドの拡張を可能にするアーキテクチャ（コンピュータ命令データフィールドをエンコードするための方法）に関する。
【０００２】
【従来の技術】ここ１０年間に渡るコンピュータアーキテクチャでは、各命令がたった１回の演算サイクル内で理想的に実行されるＲＩＳＣ（縮小命令セットコンピュータ）デバイスが、一般的になってきている。ＲＩＳＣアーキテクチュアは、標準的なアーキテクチャ及び命令セットを有するコンピュータに対して、次の点で優勢である。即ち、その点は、コンピュータが、より短い時間内に頻繁な演算の実行を可能にするために、非常に速いデータ処理速度の性能を有する点である。ＲＩＳＣデバイスは１６ビット命令セットから始まり、３２ビット命令セットアーキテクチャまで成長した。従って、ＲＩＳＣアーキテクチャのセットが、特別な特徴、例えばマルチメディア、グラフィクス、又は６４ビットデータを考慮するように、長い年月の間に増加するので、後方互換性のある方法でそのような拡張を達成することが非常に有利である。予約ビットの供給によって、将来的な拡張に対する一層の柔軟性が与えられる。
【０００３】典型的な命令セットは、即値（イミディエイト）オペランド、又はアドレスディスプレースメント（変位）を使った命令を有する。例えば、即値命令は、

算術命令のオペランドである命令のフィールド内の値を、又は、メモリをアクセスするムーブ又はロード命令用のメモリオフセットへオフセットを運ぶ。ひとたびその値のサイズが例えば8ビットに固定されたならば、値の範囲が例えば－１２８～１２７に設定される。

【０００４】

【発明が解決しようとする課題】アプリケーションが更に複雑になり、メモリサイズが成長するにつれ、後方互換性が現存の命令セットと一緒に維持されるという制約で、予約ビットを使った範囲を拡張する要求がある。

【０００５】

【課題を解決するための手段】概して、本発明は、複数のフィールドを有する第１コンピュータ命令でデータフィールドの値を拡張する方法に関連される。命令内に任意の未使用データビットが存在するか否かを決定し、もし存在するならば、未使用ビットの全て又はいくつかを用いて、データフィールド内に含まれる値を拡張する。拡張データフィールドを用いたコンピュータ命令は、未拡張データフィールドを用いたコンピュータ命令と後方互換である。

【０００６】本発明の一態様は、複数のフィールドを有し、演算コード（「オプコード」）用のフィールドと、定数、オフセット値、又は、任意の他の値である情報フィールドとを含んだコンピュータ命令で、情報フィールドを拡張する方法に関連される。また、命令は、予約ビットを含んだ予約フィールドを有する場合もある。本発明によれば、予約フィールドのビットは、情報フィールドと組合されて拡張フィールドを形成する；それから、拡張フィールドは、オプコードによって示された演算で用いられる。情報及び予約フィールドを組合せることが、Ｎビット（符号ビットを含む）の情報フィールドを、Ｎ＋Ｍビットの第１記憶位置にロードし、Ｍビットの予約フィールドをＮ＋Ｍビットの第２記憶位置にロードすることを含む場合がある。Ｎ及びＭは整数である。第１記憶位置内の情報フィールドは符号拡張される。次に、第２記憶位置はＮビットだけ左にシフトされ、右端でゼロにシフトされる。そして、第１記憶位置は、第２記憶位置と一緒にビットワイズで排他的論理和（bitwise exclusive OR'd）（ＸＯＲ）をとられて拡張フィールドを引き出す。その結果は、後方互換な拡張フィールドである。

【０００７】本発明のこれら及び他の利点と特徴とは、添付図面と共に取り上げられることになる以下の詳細な説明によって当業者に明らかになるであろう。

【０００８】

【発明の実施の形態】本発明における一実施の形態では、即値命令の符号なしオペランドが、命令によって同様に運ばれた予約ビットを用いて拡張され得る。例えば、１０ビット符号なし即値オペランドが、１予約ビットによって拡張されたと考える。予約ビットは、エンコーダによってゼロになるように要求され、もしそうでなければ命令はトラップする（割り込む）。予約ビットは、即値オペランドにおける将来の拡張ビット内の「１」になる場合もある。従って、即値範囲「０ｘｘｘｘｘｘｘｘｘｘ」、つまり０～１０２３の範囲が、「１ｘｘｘｘｘｘｘｘｘｘ」、つまり１０２４～２０４７の範囲になる場合もある。符号なし数の予約ビットにおけるこの使用は、予約ビットの任意数に適用され得る。

【０００９】命令が符号付き即値オペランドを用いた時、暗黙の符号拡張は、即値オペランド内の値が自然符号付き範囲内に拡張されるように、配列される必要がある。本来の即値フィールドにおける最上位ビットは、現存するネガティブエンコードと後方互換性を確保するために、符号拡張されなければならない。これを達成するための一例は、本来の符号拡張の即値を取り出し　即値とＸＯＲとの最終ビットを丁度通り過ぎた予約ビットを左にシフトすることである。予約ビットが現存するバイナリ内でゼロであるから、ＸＯＲは、現存する符号付き即値に影響を与えない。しかしながら、予約ビットが１に設定された時、符号付き即値の範囲における正確な拡張が存在する。例えば、１０ビット符号付き即値の使用：１０ビットの即値範囲は、「０ｘｘｘｘｘｘｘｘｘｘ」、つまり０～５１１の範囲であり、又は「１ｘｘｘｘｘｘｘｘｘ」、つまり－５１２～－１の範囲である。付加的な予約ビットで、その範囲は「０１ｘｘｘｘｘｘｘｘｘ」、つまり５１２～１０２３と、「１１ｘｘｘｘｘｘｘｘｘ」、つまり－１０２４～－５１３とに拡張される。「０ｘｘｘｘｘｘｘｘｘｘ」の場合に、セット予約ビット、つまり「１」のＸＯＲが５１２に符号付き値を加える。「１ｘｘｘｘｘｘｘｘｘｘ」の場合に、セット予約ビットのＸＯＲは符号付き値から５１２を減じる。これにより、所望の増加が範囲内で与えられる。この技術は、予約ビットの任意数に適用され、アーキテクチャの寿命の点で何倍にも適用さえできる。

【００１０】図１は、本発明の予約ビットを用いて、定数を拡張した特定の実施の形態を図示的に示す。本発明の好ましい実施の形態は、６４ビットアーキテクチャ（即ち汎用レジスタが６４ビット幅である）を有するマイクロコンピュータに組み込まれるが、初期設計の３２ビットアーキテクチャ（即ち３２ビット幅のレジスタ）に後方互換性を提供する，本発明の理解を過度に複雑にしないために、本発明の動作を記載する際に、レジスタがより小さいものと仮定する。図１では、即値命令（図示せず）に含まれる即値１１０が示される。図１は、ビットａ、ｂ、ｃ、ｄ、ｅ、ｆ、ｇから成る定数１１２と、値「ｓ」の符号ビット１１４とを含むような即値１１０を示す。オペランドが１３ビットレジスタの下位ビット内にロードされると仮定する。符号ビット１１４は、当然ではあるが、上位５ビット１１６を通して拡張されるであろう。この例では、命令によって搬送され、

「0」と「1」との種々の組合せを表す値u、v、w、x、yを有する5つの予約ビット120が存在する。予約ビット120は、別のレジスタの下位5ビット内にロードされ、そして、左に（最上位ビットに向かって上方に）シフトされてオペランドにおける符号ビットで対応するビット位置の直ぐ左に位置させる（即ち、オペランド及び符号がビット位置0～7内に存在する場合には、ビット位置8である）。予約ビット120が符号拡張ビット116と一緒にXOR124（排他的論理和）され、拡張されたオペランド値150内における5ビットの結果、即ちh、i、j、k、l126を生成する。その全結果は、オリジナルな定数値、即ち値110のビットa～gと、オリジナルな符号ビット114とから成る拡張された値150である。従って、拡張された値150は、符号ビット「s」114と、定数112と、XOR演算によって生成された新しい5ビットi、j、k、l、h126を含む。拡張された値150は、もし全ての予約ビット120がゼロであるならば、符号拡張された値110と同じである。この場合には、ビットh、i、j、k、lは符号ビット114のコンテンツを有することができる。異なるバイナリ組合せと一緒に予約ビット120を変更することによって、値110の範囲を増加できる。

【0011】図2は、予約ビットが定数の値に影響を与えない状況を示す。図2は、11ビットの汎用記憶位置210が、即値命令によって、その命令により運ばれた8ビット符号付き値（ビットp、q、r、s、t、vと「0」符号ビット）にロードされる。符号は、レジスタにおける残りの上位3ビットに拡張される。ここで、その命令は、全て「0」である3つの予約ビット212を運ぶ。図1の例に関して記載されるように、予約ビットは別のレジスタにロードされ、位置決めされるので、レジスタ記憶位置210の最も左側の3ビットと一緒に、即ちレジスタ210内にロードされた値の符号の拡張と一緒にXOR（排他的論理和）される。XOR（排他的論理和）することの結果が、レジスタ216内の記憶位置によって与えられ、図に示されるように、レジスタ210内にロードされ（且つ符号拡張され）たものと変わらない。従って、記号214の値は記号210の値と同じである。引き続き、11ビットレジスタ220が、命令によって、8つの下位位置内における2つの補数形で負の即値と、残った3つの上位ビット位置内の上方へ拡張された符号とに、現在、ロードされている。再び上記のように、命令は3つの予約ビット220を運ぶ。その予約ビットは、別のレジスタ内にロードされ、シフトされてレジスタ220の符号拡張に対応するビット位置内に位置決めされる。レジスタ220の上位ビットがそのように配置されるので、記号220の記憶位置における最も左側の3ビットが予約ビット222と一緒にXOR（排他的論理和）された時、生じたビット226は3ビットである。従って、記憶位置224の値は、記憶位置220の値と同じである。予約ビットがゼロであり、この特定の実施の形態では、それがアーキテクチャによって設定された時、最も左側の3ビットと一緒に予約ビットをXOR（排他的論理和）することがその値に影響を与えないことを、図2は示している。

【0012】図3は、予約ビットが本発明によってセットされた時に、特定の実施の形態における結果を示す。図3では、記憶位置310は、符号を示す最上位ビット311と一緒に正のバイナリ数を含む。この場合には、符号311は正、即ち「0」である。記憶位置314は、最も左側のビット315が符号ビット「1」を有する8ビットの負の数を示す。この例では、値320，322，324，326を有する2つの予約ビットが存在すると仮定する。例えば、記憶位置310が2ビット312だけ符号拡張されるとする。もし符号ビット312が予約ビット320と一緒にXORされたならば（ステップ330）、XORの結果が記号342、即ち「00」であり、そして位置340の範囲が0～127である場合に、記憶位置340の値が得られる。次に、もし記号326の予約ビット「11」が記憶位置314の符号拡張ビット316と一緒にXORされたならば（ステップ332）、その結果は、+128～+255の範囲における位置344内の「00」345である。それから、ステップ334において、予約ビット「01」322が、位置310の符号拡張ビット312と一緒にXORされる。これにより、記憶位置346内の結果「01」347が与えられ、記憶位置346が正の範囲256～383を与える。最後に、ステップ336では、予約ビット「10」324が、位置314内の符号拡張ビット316と一緒にXORされ、384～511の正の範囲を備えた位置348内の「01」349を与える。従って、位置310内における0～+127の正の範囲は、2つの予約ビットを用いて0～511まで拡張されている。また、「00」である予約ビット320によって示されるように、この演算は後方互換性があり、即ち位置340内で拡張された値は位置310内と同じ値である。

【0013】記憶位置314内の-128～-1の範囲を拡張するために、同様の手順が行われる。最も左側のビットは、位置360，362，364，366内の拡張された負の数で符号ビット「1」である。位置360の2ビット361に対して、予約ビット324は、位置310の符号拡張ビット312と一緒にXORされる。2つのビット363は、ビット316と一緒にXORされた2つのビット322からである。ビット365は、ビット312と一緒にXORされたビット326である。そして、ビット367は、ビット320と一緒にXORされたビット312である。従って、負の範囲13は-512～-1に拡張される。また、「00」である

予約ビット３２０によって示されるように、この演算は後方互換性である。即ち、位置３６６内で拡張された値は位置３１４内の値と同じである。

【００１４】図４は、本発明における特定の実施の形態を簡略化した流れ図を示す。図４ではステップ４１０において、Ｎビット即値フィールドは、長さの点で少なくともＮ＋Ｍビットである位置Ａ内にロードされる。ここに、Ｎ及びＭは整数である。即値フィールドは、ステップ４１２内で拡張された符号である。Ｍビット予約フィールドはステップ４１４内で位置Ｂにロードされる。位置Ｂの長さは少なくともＮ＋Ｍビットである。ステップ４１６において位置Ｂが、Ｎビットだけ左にシフトされ、右をゼロで満たす。ステップ４１８では、位置Ａが、位置Ｂと一緒にビットワイズでＸＯＲされ、その結果を得る。その結果は、Ｎ＋ＭビットからＲビット４２０まで符号拡張される。ここに、Ｒは整数であり、Ｒ≧Ｎ＋Ｍである。その結果の符号ビットは、Ｍフィールドの最も高いビットと一緒にＸＯＲされたＮフィールドの最上位ビットであるので、別の実施の形態では、これが計算され、最終結果においてエンコードされていない上位の全てのビットに提供されることもできる。従って、エンコードされた範囲は、[－２＾（Ｎ＋Ｍ－１）、＋２＾（Ｎ＋Ｍ－１）]である。別の実施の形態では、その位置さは、長さ６４ビットである汎用レジスタであるかもしれない。

【００１５】特定の実施形態では、図４に対する予約アルゴリズムは、ソフトウエア内で実現され、アセンブラにより利用され、拡張されたフィールドを使った命令セットをエンコードできる。Ｎビット即値範囲を拡張するために、Ｍビット予約フィールドを用いた変更アルゴリズムは：

１．範囲[－２＾（Ｎ＋Ｍ－１）、＋２＾（Ｎ＋Ｍ－１）]内のＲビット数を取り出し、
２．ステップ１の下位Ｎビットを選択し、即値フィールドとして格納し、
３．ＮビットからＮ＋Ｍビットまでステップ１の数を符号拡張し、
４．ステップ１の結果とステップ３の結果とをＸＯＲし、
５．ステップ４の結果をＮビットだけ右にシフトし（充填値は「don' t care」である）、
６．ステップ５の下位Ｍビットが予約フィールド内に格納される。

【００１６】２つの例を与える表１及び表２は、予約ビット「ｒ」を使用して６４ビットレジスタ「ｄ」内にロードされた１６ビット定数「ｓ」を拡張するか、又は１０ビットアドレスディスプレースメント（変位）「ｓ」を拡張するかのいずれかであった。

【００１７】表１は、本発明における特定の実施の形態内でのＭＯＶＩ命令の詳細な説明を示す。マイクロプロセッサは６４ビットレジスタを有し、命令の長さは３２ビットある。オプコード「１１００１１」を有するＭＯＶＩ命令は、１６ビット即値フィールド「ｓ」を符号拡張し、レジスタ「ｄ」即ちR_d内にその結果を格納する。「ｒ」フィールドは４つの予約ビット用であるかもしれない。予約ビットを用いて、Ｍ＝４及びＮ＝１６の場合に、図４を用いて１６ビット即値フィールドを拡張することもできる。異なるオプコードが必要になる場合もある。

【００１８】
【表１】

【表１】

表１

110011	s	d	r
31 26	25 10	9 4	3 0

MOVIs, Rd

source ← SignExtend(s);
Rd ← source;
Rd ← Register(result);

【００１９】表２は、本発明における特定の実施の形態内でのＬＤ.Ｌ命令（オプコード１００００）の詳細な説明を示す。ロングワード（３２ビット）は、第１ソースオペランド、即ちレジスタ「ｍ」のコンテンツを、４により位取りされた（２ビットだけ左にシフトされた）１０ビット即値に、つまり「ｓ」に加えることによって形成された有効アドレスからロードされ、即ちＬＤ.Ｌである。ロングワードは結果レジスタ、即ち「ｄ」内で符号拡張される。「ｒ」によって示された４つの予約ビット０～３がある。予約ビットを用いて、Ｍ＝４及びＮ＝１０の場合に、図４を用いて１０ビット即値を拡張することもできる。異なるオプコードが必要になる場合もある。

【００２０】
【表２】

【表2】

表2

100010	m	s	d	r
31 26	25 20	19 10	9 4	3 0

LD.L Rm, s, Rd
base ← ZeroExtend64(Rm);
offset ← SignExtend10(s) << 2;
address ← ZeroExtend64(base + offset);
result ← SignExtend32(ReadMemory32(address));
Rd ← Register(result);

【００２１】ハードウエア具体化における特定の実施の形態

特定の実施の形態では、本発明は、ゼロ又はルート階層レベルで、Ｓ５コア５００ユニットを有するＣＰＵ内で実行される。コンピュータコア５００は、階層レベル１で６つのユニットを有し、オプションで着脱自在な浮動小数点ユニット（ＦＰＵ）を含む。図５は、本発明におけるコンピュータコア５００の一例の簡略化ブロック図を示す。表３はＳ５コア内の各ユニットの機能を示す。命令フローユニット（ＩＦＵ）２１０は、整数命令を処理することを含み、更に付録１で記載される。

【００２２】

【表３】

【表3】

表3

階層レベル	ユニット名	略称	説明
0	Ｓ５コア５００	S5	最上位レベルコアブロック
1	バスインタフェースユニット５０５	BIU	周辺モジュールのような外部モジュールと外部メモリインタフェースとにバスアクセスを制御する。
1	命令フローユニット５１０	IFU	ＣＰＵパイプのフロントエンド：フェッチと、デコーダと、イシュウ及び分岐。またモードＢエミュレーションを含む。
1	整数／マルチメディアユニット５２０	IMU	全ての整数とマルチメディア命令とを取り扱う。メインＣＰＵデータパス。
1	命令キャッシュユニット５３０	ICU	命令キャッシュと命令変換索引バッファ（ＴＬＢ）とを比較する。
1	ロード／ストアユニット５４０	LSU	全てのメモリ命令とデータキャッシュコントロールとを取り扱う。
1	データキャッシュユニット５５０	DCU	データキャッシュとデータ変換検索バッファ（ＴＬＢ）とを比較する。
1	浮動小数点ユニット（図示せず）	FPU	着脱自在な浮動小数点ユニット、パイプ制御、及び実行パイプ（図５に図示せず）。

【００２３】図６は、整数／マルチメディアユニット（ＩＭＵ）５２０の一例のブロック図を示す。特定の実施の形態では、ＩＭＵ５２０はｃｐｕ算術命令を取り扱い、整数、マルチメディア演算、及び、論理命令を含む；ロード／ストアアドレス計算と範囲外（アドレス誤り）検出；分岐と一部分岐との比較；及び分岐ターゲットアドレス計算。ＩＭＵ５２０の計算は、第１（ｅｘｅ１）、第２（ｅｘｅ２）、及び、第３（ｅｘｅ３）パイプラインステージの間に生じるかもしれない。サブユニットの多くは１サイクルの実行時間を有するが、乗算器は３サイクルの待ち時間を有する。この実施の形態のＩＭＵ５２０はシンプルなパイプラインである。もしｉｆｕ＿ｉｍｕ＿ｓｔａｌｌ信号、又はｉｆｕ＿ｉｍｕ＿ｉｎｖａｌｉｄａｔｅ信号が受信されなければ、従属信号により遮断されることなく、データが各サイクルをパイプライン内で進む。ＩＭＵ５２０は、ＩＦＵ５１０からその入力の多くを取得する。メイン信号はソースオペランドとオプコードワードとである。ＩＭＵ５２０は、ｅｘｅ１及びｅｘｅ２パイプラインステージ（１又は２サイクル命令）、又は、ｅｘｅ３パイプラインステージ（３又は４サイクル命令）でその結果を送ることができる。

【００２４】特定の実施の形態では、ＩＭＵ５２０は７つのユニットを有する。表４は図６内のＩＭＵブロックの説明を与える。「AN INTEGER INSTRUCTION SET ARCHITECTURE AND IMPLEMENTATION」という名称の同時出願の

米国特許出願第09／410，683号（弁護士事件番号第16869A-0003700US）は、全ての目的のためにそっくりそのまま本出願の一部分に含まれ、

【表4】

表4

階層レベル	ユニット機能	略称	説明
2	加算器	ADA 616	整数及びアドレス加算器
2	加算器	IMA 620	simd 加算器
2	セレクタ	XSL 630	exe1 exe2 exe3 の結果マクス
2	エクストラハードウエア	XHW 640	特定の命令のサポート
2	シフタ／シャフル	SHF 650	シフタ及びシャフル
2	乗算器	MUL 660	整数及び simd 乗算器
2	制御	XCT 670	制御ブロック

【００２６】例えば、ＡＤＡ６１６は、ＩＦＵ５１０又はＬＳＵ５４０のアドレス指定と、整数加算、減算又は比較とを行うことができる。ＡＤＡ６１６は６４ビット加算器を備えることもできる。第２入力は減算又は比較用に反転される。範囲検査をも行うことができる，

【００２７】例えば、ＩＭＡ６２０は２つのアドレスを含み、各アドレスが１つの３２ビット加算、２つの１６ビット加算、又は、４つの８ビット加算を実行する。各タイプの加算は、内部に桁上げを含み、全ての加算に対して同時に起動される。その結果が８ビットベクトルになる場合には、これによってバイト処理が可能となるだろう。第２ソースオペランドは、減算及び比較を許容するように、反転されることもできる。

【００２８】特定の実施の形態におけるＸＨＷ６４０は、他のブロックのハードウエアを再利用することによって実行することが困難な命令を取り扱う。その命令は実行用の自らのハードウエアを有し、論理ＡＮＤ、ＯＲ、ＡＮＤ　Ｃ、及び、ＸＯＲと、ＭＯＶＩ、ＳＨＯＲＩ、及び、ＮＳＢ命令とを含むことができる。

【００２９】例えば、ＳＨＦ６５０は、６４ビットソース内のバイト量のディスプレースメント（変位）を含んだ命令だけでなく、シフト関連命令を実行することができる。それは、飽和時にイシュウ（発行）された１サイクル遅れ（ｅｘｅ２）を除いて、１サイクル（ｅｘｅ１）内にその命令の多くを実行する。また、それは、もし飽和時のコマンドがｅｘｅ２で生成されたならば、その飽和値をマクス（多重化）する。

【００３０】シャフル及びバイトディスプレースメント機能も同様に、３ステップ内でＳＨＦ６５０によって実行されることが好ましい，第１ステップは、シャフルマトリクス用の制御信号を作成する。第２ステップはバイトを移動する。最後のステップは、もしｅｘｅ３で必要とされるならば、飽和値をマクスする。

【００３１】飽和検出は、ＳＨＦ６５０によって個別的に行われる。それは、シフトパス（経路）からのオペランドとマスクとを使用し、シフト及びシャフルの第２サイクル（ｅｘｅ２）のために飽和コマンドを生成することができる。

【００３２】例えば、ＭＵＬ機能６６０は、乗算関連命令を実行し、ＳＩＭＤ（単一命令多重データ）整数乗算を含む。命令の多くは３サイクル内で終了するが、いくつかの命令は、終了するまでに４サイクルを必要とする。ブースエンコードアルゴリズム（Booth encoding algorithm）を用いて、乗算を実行できる。

【００３３】ＸＣＴ６７０は、ＩＦＵ５１０によって供給される種々の制御信号をバッファに保留し、且つ広めるために動作できる。その種々の制御信号は各実行ブロック用に再現できるために必要とされる。また、それは、パイプ内での命令位置の小さな表を保持することによって、パイプ内でのブロックを制御することもできる。ＸＣＴ６７０は、有効、ストール（停動）、及び、無効の信号を考慮することができる。

【００３４】本発明における特定の実施の形態では、例えば、表１及び表２で与えられた命令がＩＦＵ５１０内でデコードされ、符号拡張後に、４ビットＸＯＲがデコードパスに追加されて「ｓ」の範囲を増大させる。別の実施の形態では、例えば、ＭＯＶＩ内の即値命令「ｓ」、又は、ＬＤ．Ｌ 内のディスプレースメント「ｓ」を、汎用レジスタ（ＧＰＲ）と、別のＧＰＲ内の予約ビット「ｒ」と、ＩＭＵ２５０内で達成されたシフティング（シフト処理）、ＸＯＲ論理、及び符号拡張とにロードできる。

結論

上記実施の形態では、本発明は、特定の典型的な実施の形態に関して記載されている。他の実施の形態は当業者にとって明らかであろう。例えば、命令の長さは１６ビット又は６４ビットでもよく、そしてマイクロプロセッサは１６、３２、１２６ビットバス及びワードで動作することもできる；図４のアルゴリズムは、ハードウエア、ソフトウエア、又はその両方の組み合わせにおいてダウン可能である：そして、データはリトルエンディアン又はビッグエンディアンのいずれかのフォーマットにすることもできる，従って、添付された特許請求の範囲とその等価形態の全範囲とで説明されるように、本発明

の幅広い思想及び範囲から逸脱することなく、種々の変更及び変形を実施できることは明らかであろう。

【００３５】

【発明の効果】本発明によれば、即値命令の符号なしオペランドが、命令によって同様に運ばれた予約ビットを用いて拡張され得る。符号なし数の予約ビットにおけるこの使用は、予約ビットの任意数に適用され得る。この技術は、予約ビットの任意数に適用され、アーキテクチャの寿命の点で何倍にも適用さえできる。異なるバイナリ組合せと一緒に予約ビットを変更することによって、値の範囲を増加できる。予約アルゴリズムは、ソフトウエア内で実現され、アセンブラにより利用され、拡張されたフィールドを使った命令セットをエンコードできる。

付録１：命令フローユニット（ＩＦＵ）

１－１　概観

命令フローユニット（ＩＦＵ）はＳ５コアのシーケンサとして機能する。ＩＦＵはコア内の命令及びデータの流れを調整し、且つコア内部アクティビティを伴う外部イベント（事象）をマージする。その主な機能は、命令キャッシュユニット（ＩＣＵ）から命令をフェッチし、その命令をデコードし、その相互依存性を解決し、レジスタファイルからオペランドを読み出し、デコードされた命令及びオペランドを実行ユニット（整数マルチメディアユニットＩＭＵ及びロードストアユニットＬＳＵ）に送出し、実行ユニットからその結果を回収し、そしてレジスタファイルにその結果を書き込むことである。更に、ＩＦＵは、命令キャッシュミス時に、バスインタフェースユニット（ＢＩＵ）と接続し、外部メモリからの欠落した命令で命令キャッシュを満たす。また、ＩＦＵは、外部可観測性のためにＳ５コア内部データの転送を調整するために、デバッグユニットにも接続する。

【００３６】アーキテクチャはモードＡ及びモードＢと呼ばれ、その間で切り替わるための機構を有する２つの命令セットを提供する。

【００３７】モードＡ命令セットは、固定長３２ビット符号化を用いる命令を表す。モードＡは、最適な動作が必要とされる場合に、又は、ＣＰＵ制御及び構成機構にアクセスするために用いられる。典型的には、全てのモードＡ命令は４バイト長であり、４バイト境界上のメモリ内に保持される。命令は、０～３１を付された３２ビットの集まりとして記載される。

【００３８】モードＢ命令セットは、固定長１６ビット符号化を用いる命令を表す。モードＢは、ＳＨアーキテクチャの以前の態様とのユーザモード命令レベル互換性を提供する。モードＢは、コード密度或いはＳＨ互換性が要求される場合に用いられる。

【００３９】ＩＦＵの別の主なタスクは、順序制御の役割に加えて、全てのモードＢ命令のエミュレーションを実行することである。詳細には、全てのモードＢ命令は、１つのモードＡ命令か、又は、モードＡ命令のシーケンスのいずれかに変換される。その後、これらの（モードＡ）命令は、元のモードＡ命令実行意味にわずかに変化を加えて実行される。このエミュレーションアプローチを用いる場合、モードＢ命令を実行するのに必要な論理は、数ブロック内で切り離すことができる。これは、モードＢの定義が変化した場合に、又は、モードＢ互換性に対する必要条件が欠落した場合に、少数の論理ブロックのみが影響を受け、そのブロックしか再実行される必要がないという利点を有する。

１．１．１　ブロック図

図７は、全ての内部ブロック及び全ての外部インタフェース用ユニットを有する簡単なＩＦＵブロックを示す。Ｓ５コアの順序制御機能のため、ＩＦＵは、コアに搭載された大部分の他のユニットとのインタフェースを有する。ＩＦＵとＢＩＵとの間のインタフェースは、命令キャッシュへの命令のロードを開始する。ＩＦＵとＩＣＵとの間のインタフェースは、実行のための命令のフローを提供する。ＩＦＵとＩＭＵ／ＬＳＵとの間のインタフェースは、命令、オペランド、結果及び命令を実行可能にするための制御信号を送受信するための経路を提供する。ＩＦＵとデバッグユニットとの間のインタフェースは、Ｓ５コアと外部デバッグ用エージェントとの間で、サンプリングコマンド、サンプリングされたデータ及び他のデバッグコマンドを交換するのを容易にする。これらのインタフェースに加えて、ＩＦＵは、外部割込みをサンプリングし、且つ、仲介する外部割込みコントローラから外部割込み信号を受信する。そのＩＦＵは、内部例外でその外部割込みを仲介し、非同期イベントを処理するために適当なハンドラを起動する。

【００４０】内部的には、ＩＦＵは、その機能に従ってブロック、即ち命令キャッシュ制御ユニット、フェッチユニット、分岐ユニット、デコードユニット、パイプ制御ユニット、及び、オペランドファイルユニットに分割されることが可能である。表５は、これらのブロック及びその頭文字を掲載する。

【００４１】

【表５】

【表5】

表5

階層	ブロック名	略称
2	命令キャッシュ制御ユニット	ＩＣＣ
2	フェッチキャッシュ	ＦＥ
2	分岐ユニット	ＢＲ
2	デコードユニット	ＤＥＣ
2	パイプ制御ユニット	ＰＰＣ
2	オペランドファイルユニット	ＯＦ

【００４２】１．１．２　ＩＦＵブロックの機能の説明
・命令キャッシュコントロール（ＩＣＣ）
ＩＣＣは、命令キャッシュへのアクセスをセットアップするために、ＦＥと内部的に、且つＩＣＵと外部的に通信する。標準的には、ＦＥは、命令フェッチアドレスと、ＩＣＣへの「フェッチ要求」を示す１組の制御信号とを供給する。その代わりに、ＩＣＣはＦＥに、２ワードに整列した命令ワードを送出する。命令キャッシュが欠落する場合に、ＢＩＵに対する再充填サイクルを開始し、外部メモリから欠落しているキャッシュラインをロードする。再充填は、ＦＥが元のフェッチアドレスに存続している間に生じる。別の方法では、ＦＥは、命令を返送する必要がない「プリフェッチ要求」を与えるか、又は、キャッシュが欠落する場合に再充填を起動する必要がない「フェッチ要求」を与える。これらの異なるタイプの要求を取り扱うために、ＦＥとＩＣＣとの間で１組のプロトコルが定義される。
・フェッチユニット（ＦＥ）
命令がキャッシュからフェッチされる場合、その命令は１組の４つの命令バッファに保管されるか、又は、ターゲットアドレスレジスタに関連するバッファ空間（Ｔ０－Ｔ７、即ちＩＡＲ）に格納されるようになる。厳密には命令が格納される場所は、命令のＦＥ及びＩＳＡモードによりその命令がどのように要求されるかによる。しかしながら、最終的には、２つの命令バッファのうち１つ（モードＡデコード器の場合のｉｂ０、及び、モードＢデコード器の場合のｉｂ２）にシフト／移動し、その中でデコードされ、実行ユニットに送出されるであろう。これらのバッファ間での命令のシフト／移動は、これらのバッファ空間の使用率を最適化しようとするＦＥにより制御される。
【００４３】ＦＥの別の役割は、フェッチされた命令のアドレスから順次アドレスを導出し、その命令が必要になる前にこれらの命令を「プリロード」することである。「プリロード」命令は実行されることは保証されず、それゆえ無駄な電力消費の一因となる可能性もあるため、ＦＥにより実施されるプリロード動作は、過剰な電力使用が生じるほど積極的に行うべきではない。一方、この動作は、必要な命令のフェッチの遅れに起因して、過剰な無駄が生じることもないように、あまりに少なすぎてもならない。
・デコードユニット（ＤＥＣ）
Ｓ５コアがモードＡ（ＩＳＡ＝＝１）下で実行中に、モードＡデコード器は、ｉｂ０の命令をデコードし、そのデコードした命令情報を、内部ではＦＥ、ＢＲ及びＰＰＣに、外部ではＩＭＵ及びＬＳＵに送出する。これらの情報により、ＰＰＣが、レジスタ使用ハザード（リード・アフター・ライトの真の依存性及びライト・アフター・ライトの非依存性）及び命令直列化要件を迅速に解決できるようになる。また、その情報により、ＩＭＵ及びＬＳＵは、更に命令をデコードすることなくデータ演算を開始できるようになる。分岐命令の場合、迅速な部分デコード分岐命令により、ＢＲは、分岐条件を統計的に予測できるようになり、最も早い時間に分岐を決定する可能性もある。
【００４４】Ｓ５コアがモードＢ（ＩＳＡ＝＝０）下で実行中に、全ての命令は、Ｔステージと呼ばれる、付加的なモードＢパイプラインステージを介して進む。その命令は、最初にｉｂ２まで移動する必要があり、その中でその命令は、１つ又は多数のモードＡエミュレーション用命令に変換されるようになる。その後、エミュレーション用命令はｉｂ０に移動し、その中では標準モードＡ命令の実行が再開される。いくつかの論理ブロックのある細部を除いて、Ｓ５コアの残りの部分は、これらの２つのモード間の差を知るべきではない。
・分岐ユニット（ＢＲ）
分岐ユニットは、ＩＳＡ仕様書に定義される全ての分岐に関する命令を取り扱う。分岐ユニットは、デコードされた分岐命令をＤＥＣから受信し、分岐条件及びターゲットアドレスがわかるか否かを決定し、その分岐を決定／予測し始める。分岐条件が未知である場合には、分岐命令の「１」ビットに基づいて、ＢＲは統計的に分岐条件を予測するであろう。その後、予測された命令がフェッチされ、デコードされる。ある状況では、分岐条件が決定される前に、予測された命令がフェッチされ、デコードされる場合もある。その状況が生じる場合、ＢＲがその予測を正確であると判断するまで、その予測された命令はデコードステージに保持されるであろう。
【００４５】分岐命令がデコードされた場合に、ターゲットアドレスが未知である場合には、ターゲットアドレスが使用可能になるまで、分岐命令はデコードステージに保持されるであろう。ターゲットアドレス転送の実施

制約条件に起因して、最適な分岐性能を得るために、準備－ターゲット命令と分岐命令との間には適当な「間隔」が必要とされる。

・パイプ制御ユニット（ＰＰＣ）

ひとたびＤＥＣにより命令がデコードされれば、ＰＰＣが残りのパイプステージを介してその実行状態をモニタする。ＰＰＣの主な機能は、（ｉ）必要時（ＩＭＵの乗算－累積内部転送の場合）に、全てのソースオペランドが準備状態又は準備可能な状態になるまで命令がデコードステージに保持されることになる、（ｉｉ）命令及び内部／外部イベントにより課せられた全ての同期及び直列化要件が生じる、（ｉｉｉ）全てのデータオペランド／一時的な結果が正確に転送されるという点で、命令が円滑、かつ正確に実行されるようにすることである。

【００４６】パイプコントロール論理を簡単にするために、ＰＰＣはモードＡ命令実行時に、いくつかの観測及び仮定を行う。その仮定の１つは、ＩＭＵ命令のいずれも例外を生じず、全ての命令が決定的にパイプステージを介して流れるというものである。この仮定によりＰＰＣは、ＩＭＵを、入力オペランドが到来する場所及び出力結果が送出される場所を知る必要がない複合データ演算エンジンとして見ることができるようになる。その後、全てのデータ転送及びハザード検出論理は、同じ簡単な機構を用いてＰＰＣ内に一体化させることができる。ＬＳＵパイプライン内の非決定的な演算に対して適応するために、その後、この簡単な機構にいくつかの工夫がなされる。しかしながら、その変更は、特に、ＬＳＵパイプラインの特異性においてターゲットとされており、オーバヘッドが最小限になるようにすべきである。

【００４７】ＰＰＣの別の主な機能は、命令例外、外部割込み、リセット等のような非順次イベントを取り扱うことである。正常な例外条件下では、ＰＰＣの一部は常時アイドル状態にある。イベントが生じる際に、ＰＰＣが起動する。ＰＰＣは外部割込みコントローラから外部割込み／リセット信号を受信する。ＰＰＣはコアの多数の部分から内部例外を受信する。いずれの場合でも、ＰＰＣはパイプラインを一掃し、その後、ＢＲにコア状態を保管することを通知し、適当なハンドラに分岐するであろう。多数の例外及び割込みが同時に生じる場合に、構造的に定義された優先順位に従って、その間で仲介する。またＰＰＣは、ＳＲ．ＢＬビットを確認して、その割込み／例外が妨げられるべきか否か、及びそのタイミングを知る。

・オペランドファイル（ＯＦ）

ＯＦユニットは構造的に定義された汎用レジスタファイルを実現する。さらに、まだ完遂されていない一時的な結果を格納し、かつ転送するために「パイプファイル」と呼ばれる、限定形のリオーダバッファも実現する。Ｓ５コアにより採用されたｉｎ－ｏｒｄｅｒ（順次）実行の性質に起因して、実行結果がｏｕｔ－ｏｆ－ｏｒｄｅｒ（非順次）に生成される場合もある小さな時間のウインドウのみが存在する。Ｓ５コアはこの特性を利用して、結果が生成された直後に一時的な結果が転送されるようにできる、簡略化した形のリオーダバッファを実現すると共に、通常リオーダバッファに関連する高コストのタグ移動／一致機構をなしで済ませる。ＯＦは、このパイプファイルのデータ経路部分を実現する。制御はＰＰＣにおいて実行される。

１．１．３　ＩＦＵパイプライン構造

基本的なパイプラインフローを理解するために、以下の図には２つの一般的な場合が示される。

【００４８】図８は、データ依存性もなく、データ間のリソース依存性もない命令シーケンスを示す。その命令は、パイプライン内に空き（babble）がなく実行される。

【００４９】図９は、連続した（back-to-back）依存性を有するパイプラインフローを示す。これは、全てのＩＭＵの１サイクル実行命令に当てはまる。その結果は、以下の命令に即時利用可能である。その依存性はパイプラインに空きを引き起こさないであろう。

【００５０】この節は、図１０及び図１１に示されるＩＦＵの観点から見たパイプライン構造を議論する。フロントエンドパイプステージは、１．５サイクルの実行キャッシュアクセス時間と０．５サイクルの事前デコードステージとに起因して、わずかに異なって言及されることに注意されたい。・フェッチステージ（Ｆ）フェッチステージは１．５クロックサイクル（又は３クロックフェーズ）間続く。Ｆステージ中に、ＩＣＣは、命令キャッシュにアクセスし、ＦＥに第３クロックフェーズの開始時点でキャッシュヒット／ミスについて通知し、ヒットの場合には、そのフェーズの最後にフェッチされた命令を戻す。ＦＥはＦステージの１サイクル中にフェッチアドレスをインクリメントする。フェッチアドレスが奇数ワード境界上に位置する場合には、４だけインクリメントされ、そのインクリメントされたアドレスが同じサイクル中にＩＣＣに送出され、後続のサイクル（即ち、現在のＦステージにおける第３のフェーズにおいて開始するサイクル）において命令キャッシュにアクセスする。フェッチアドレスが偶数ワード境界に位置する場合には、８だけインクリメントされ、そのインクリメントされたアドレスが、ＩＣＣに送出される前に１サイクル間ラッチされる。直後のサイクルの場合、ＦＥは、ＩＡＲプリロード或いは命令キャッシュプリフェッチのような優先順位の低いキャッシュアクセスでそのサイクルを満たす。

【００５１】Ｆステージは１．５サイクル続くが、命令キャッシュは、各サイクルに命令キャッシュを取り扱うことができるようにパイプライン処理される。その結果、ＦＥはサイクル毎に動作する必要がある。

・事前デコードステージ（Ｐ）

命令がフェッチされた直後に、その命令は（事前）デコード論理に給送され、その中でＤＥＣが迅速に、（ｉ）３レジスタファイル（ＲＦ）読出しポートのうち任意のポートにアクセスする必要があるか否かを、及び、その命令が、ターゲットアドレスレジスタ（ＩＡＲ．Ａ）とそれに関連するバッファ空間（ＩＡＲ．Ｔ）とに読み出される必要がある分岐命令であるか否かを判定する必要がある。この事前デコード処理は、ＲＦ、ＩＡＲ．Ａ及びＩＡＲ．Ｔが全て特別に作製したアレイとして実装され、読み出しを開始するためにきれいなクロックエッジを必要とするため、Ｄステージに対するクロックの立ち上がり前に行われなければならない。これらの２つのタスクに加えて、他のデコードステージタスクは、タイミング要件を満足するためにデコードステージからこのステージに移行される場合もある。

・デコードステージ（Ｄ）

このステージの間に、ＤＥＣ及びＰＰＣがともに命令をデコードし、あらゆる直列化／例外／機能停止条件を検査し、更に利用される可能性のあるソースオペランドがＲＦ、コントロールレジスタスタック、ＩＡＲ．Ａ等から検索される間に、全ての必要とされるデータ転送を取り扱うように動作する。全てが良好に進行するなら、デコードされた信号は、正確なソースオペランドとともに、実行ユニット（ＩＭＵ又はＬＳＵ）に配向され、微視的なアーキテクチャ状態が更新され、その命令が命令バッファから除去される。その命令が無条件又は分岐すると予測された命令である場合には、いずれかのターゲットアドレスが次のフェッチサイクル間にＩＡＲ．Ａから検索されるか、又は、ターゲット命令がプリロードされている場合には、その命令及び次のフェッチアドレスが、次のデコード及びフェッチステージの間にＩＡＲ．Ｔバッファから読み出される。

・変換ステージ（Ｔ）

コアがモードＢで動作している場合、全ての命令がこの変換ステージを通り、モードＡ命令に翻訳される。コアの残りの部分はサイクル当たり大部分の１（モードＡ）命令で処理することができるため、モードＢ命令が１つのモードＡ命令でエミュレートできない場合には、その変換を取り扱うために多数サイクルかかるであろう。本質的には、モードＢ命令はＴステージに位置し、一方エミュレート用のモードＡ命令は、サイクル当たり１回、Ｄステージ及びパイプラインの残りの部分に給送される。モード切替え分岐命令を除いて、ＤＥＣは単にモードＢ命令を取り扱うための役割を果たす。

・Ｅ１実行ステージ（Ｅ１）

条件付き分岐命令の場合、ＩＭＵは６４ビット比較演算を行うように命令され、Ｅ１ステージ中にＩＦＵに１ビットの結果を転送する。この結果ビットは、ＢＲ及びＤＥＣ（又はＰＰＣ）により同時に検査される。それが予測されたものである場合には、変更なく全てが進行する。そうでない場合には、ＤＥＣがＤステージの命令を無効にし、一方ＢＲが、次の（Ｅ２）ステージにおいて予測誤り修復プロセスを開始する準備をする。

・Ｅ２実行ステージ（Ｅ２）

ＢＲが前ステージ中に予測誤り分岐を検出する場合には、Ｅ２サイクル中に正確な経路にそのフローを再配向する。実際の分岐方向及び正確な次の命令がフェッチされたか否かにより、いくつかのバブルがパイプラインに挿入されるようになる。

【００５２】ＰＴ命令の場合、Ｅ１サイクル中にＩＭＵアドレス加算器により計算された、ターゲットアドレスレジスタのための新しい内容は、ＯＦのパイプラインを介してこのサイクル中にＢＲに転送され、Ｗステージ中にターゲットアドレスレジスタファイル（ＩＡＲ．Ａ）に書き込まれるはずの、パイプラインに類似のステージングキューにラッチされる。しかしながらこの新しい内容は、即時にＦＥに転送され、ＦＥが次のサイクル中にターゲット命令をプリロードできるようにする。

・Ｅ３実行ステージ（Ｅ３）

このサイクル中に、ＩＣＣがＰＴ命令のターゲットをプリロードするためにＩキャッシュ（ＩＣａｃｈｅ）へのアクセスを開始することができる。ターゲットプリロードは性能の最適化のためであるため、より高い優先順位のフェッチ要求が存在する場合には、プリロードは生じない場合もある。他の２つの命令、ＩＣＢＩ及びＰＲＥＦＩは、ＩＣＣ（及び、従ってＩキャッシュ）に対する異なる要求タイプを除いて、同様に処理されることに注意されたい。

【００５３】また、任意の例外条件が生じたか否かを決定することが、このステージにおいて生じる。Ｅ３サイクルの終了前に、ＰＰＣは全ての発生する可能性のある例外条件を検査し、内部フラグを立て、ライトバックステージにおいて即時にハンドラ開始シーケンスを開始できるようにする。

・ライトバックステージ（Ｗ）

Ｗステージ中に、例外条件が検出されない場合には、アーキテクチャ状態が更新される。これは、ＢＲにおけるレジスタファイル（ＲＦ）、ターゲットアドレスレジスタファイル（ＩＡＲ．Ａ）、及び、コントロールレジスタを含む。例外条件が検出された場合には、ＰＰＣがＢＲにハンドラ開始シーケンスを開始したことを通知する。

・開始ステージ（Ｌ）

例外ハンドラ開始シーケンスは数サイクル間続く。

【図面の簡単な説明】

【図１】本発明の予約ビットを使った定数を拡張する特定の実施の形態を示す図である。

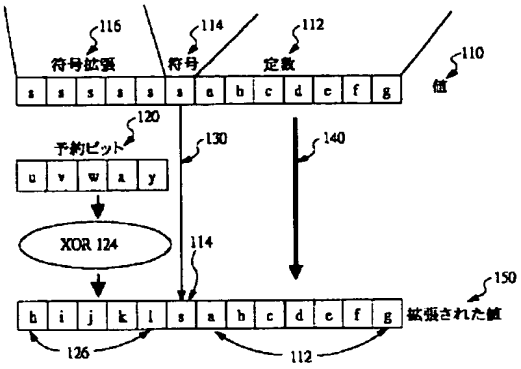
【図２】予約ビットが本発明の定数の値に影響を与えない場合において、特定の実施の形態を示す図である。

【図３】予約ビットが設定された時に、本発明における

特定の実施の形態の結果を示す図である。
【図４】本発明における特定の実施の形態の簡単な流れ図である。
【図５】本発明のコンピュータコアの一例を示す簡単なブロック図である。
【図６】整数／マルチメディアユニットの一例を示すブロック図である。
【図７】付録１において、全ての内部ブロック及び全ての外部インタフェース用ユニットを有する簡単なＩＦＵブロック図を示す。
【図８】付録１において、データ依存性もなく、データ間のリソース依存性もない命令シーケンスを示す図である。
【図９】付録１において連続した依存性を有するパイプラインフローを示す図である。
【図１０】付録１においてモードＡのＩＦＵパイプラインを示す図である。
【図１１】付録１においてモードＢのＩＦＵパイプラインを示す図である。
【符号の説明】
１１０　値
１１２　定数
１１４　符号
１１６　符号拡張
１２０　予約ビット
１２４　ＸＯＲ
１２６　ビット結果
１５０　拡張された値
２１０　記憶位置（レジスタ）
２１２　予約ビット
２１４　記憶位置（レジスタ）
２１６　レジスタ
２２０　記憶位置（レジスタ）
２２２　予約ビット
２２４　記憶位置（レジスタ）
２２６　結果ビット
３１０　記憶位置
３１１　最上位ビット
３１２　２ビット
３１４　記憶装置
３１５　ビット
３１６　符号拡張ビット
３２０　予約ビット
３２２　予約ビット
３２４　予約ビット
３２６　予約ビット
３３０　ステップ
３３２　ステップ
３３４　ステップ
３３６　ステップ
３４０　記憶位置
３４２　２ビット
３４４　記憶位置
３４５　２ビット
３４６　記憶位置
３４７　２ビット
３４８　記憶位置
３４９　２ビット
３６０　記憶位置
３６１　２ビット
３６２　記憶位置
３６３　２ビット
３６４　記憶位置
３６５　２ビット
３６６　記憶位置
３６７　２ビット
４１０　ステップ
４１２　ステップ
４１４　ステップ
４１６　ステップ
４１８　ステップ
４２０　ステップ
５０５　バスインタフェースユニット
５１０　命令フローユニット
５２０　整数／マルチメディアユニット
５３０　命令キャッシュユニット
５４０　ロード／ストアユニット
５５０　データキャッシュユニット
６１６　アドレス加算器（ＡＤＡ）
６２０　整数／ｓｉｍｄ加算器（ＩＭＡ）
６３０　ｅｘｅ１　ｅｘｅ２　ｅｘｅ３セレクタ（ＸＳＬ）
６４０　エクストラハードウエア（ＸＨＷ）
６５０　シフタ／シャフル（ＳＨＦ）
６６０　乗算器（ＭＵＬ）
６７０　制御（ＸＣＴ）

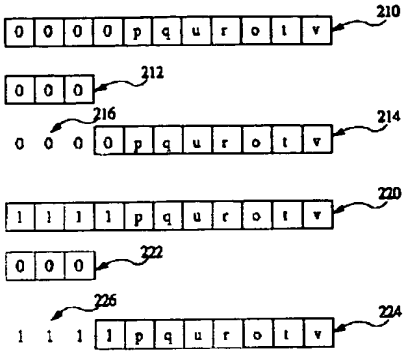
【図1】

図 1



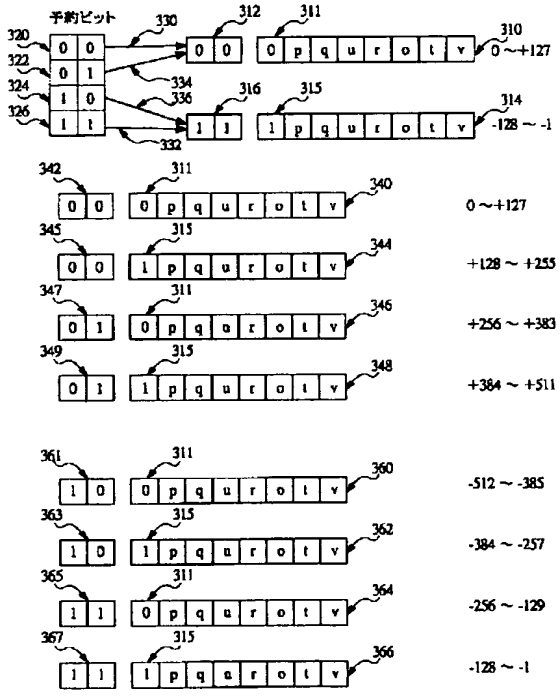
【図2】

図 2



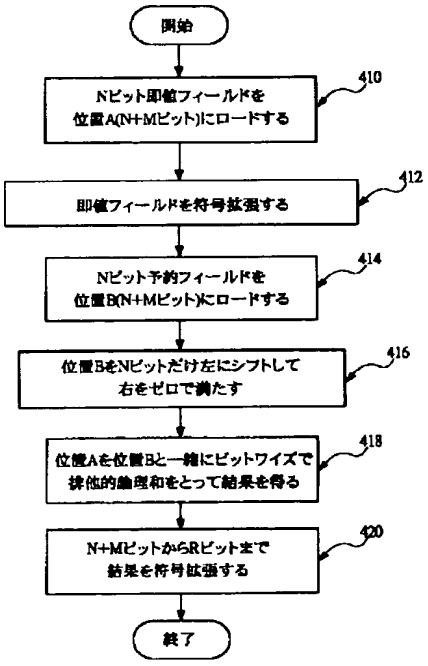
【図3】

図 3



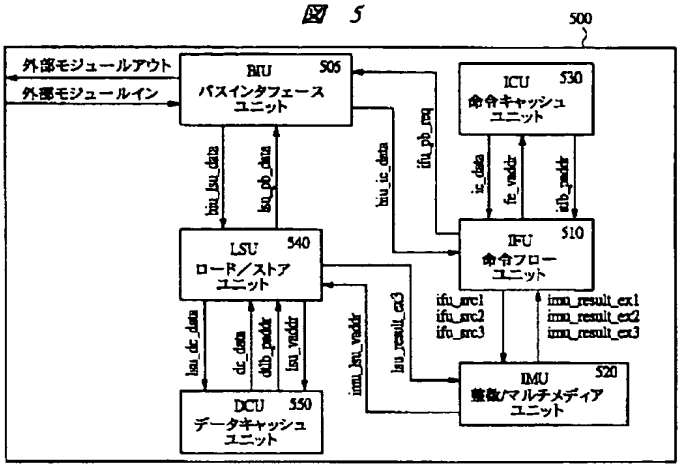
【図4】

図 4



【図5】

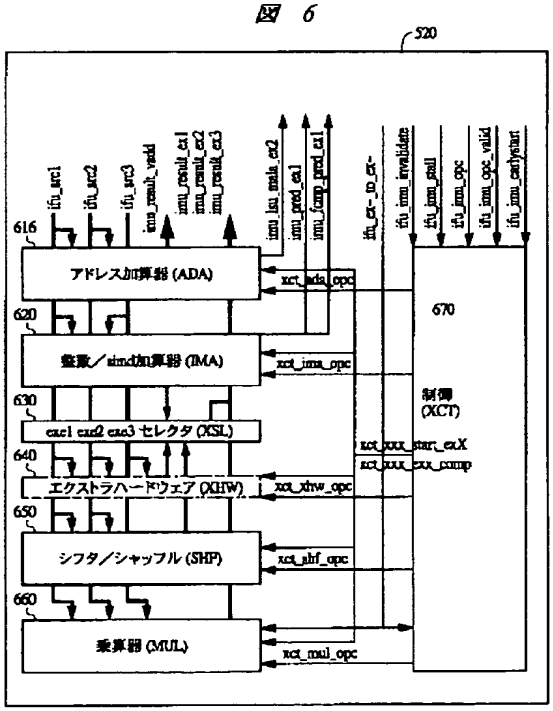
図 5



注意：LSU結果はIMUへ行き、そこでは乗算器結果と一緒にマスクされて全てのimu-result-ex3を形成する。32/64ビットバスがブロックの間で動く。

【図6】

図 6



【図7】

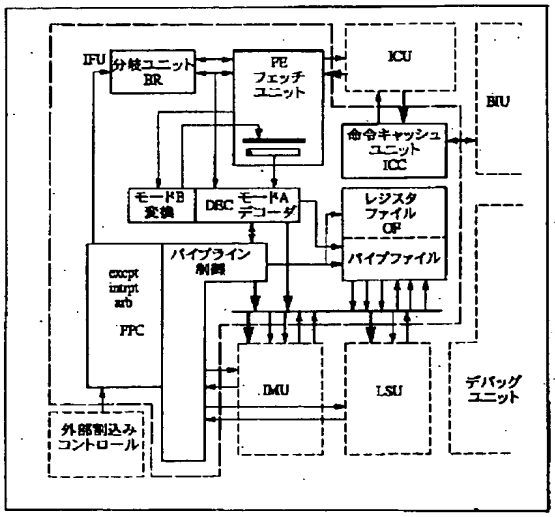


図7　IFU内部ブロック図

【図8】

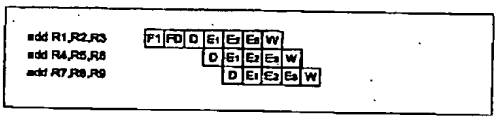


図8　データ依存性のない基本的なパイプラインフロー

【図9】

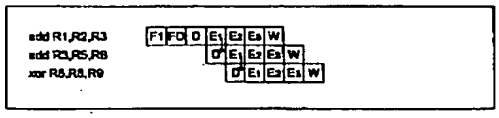


図9　連続したデータ依存性を備えたパイプラインフロー

【図10】

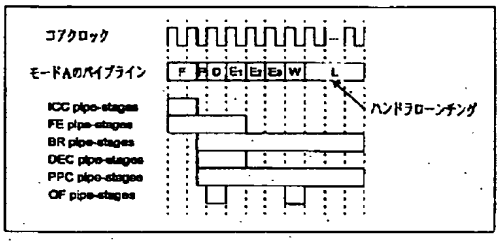


図10　モードAのＩＦＵパイプライン図

【図11】

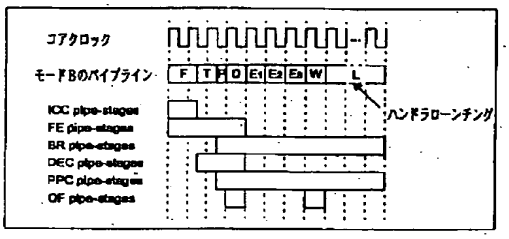


図11　モードBのＩＦＵパイプライン図